CS572 Group Project Proposal

Jinze Cui Fengyi Quan

1 Introduction

Adversary attack, a technique that tries to fool models with malformed data, is a trending topic that has been discussed heatedly in the deep learning area. However, compared with its success in Computer Vision, the related research in Natural Language Processing (NLP) is limited. Therefore, our group determines to reproduce an important adversary attacking method in NLP and conduct several further analyses about it.

We will focus on an ACL-2019 paper called *Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency* [1]. It proposes a novel attacking method called PWWS (Probability Weighted Word Saliency) to generate adversarial examples on text classification tasks. The main idea of PWWS is to substitute the words in a given text with their synonyms. It combines the word saliency and the classification probability to determine the word substitution order and changes the words greedily until it can successfully fool the NLP model. The generated adversarial examples are guaranteed to be lexical correct without grammar mistakes or semantic shifting.

Our project aims to reproduce the PWWS and use it to generate adversary examples. We will also implement other attacking methods as baselines. The basic ideas of these methods are introduced in Section 2. We will use these methods to attack the state-of-the-art NLP models on text classification. The performance of the models on the adversary examples will indicate the efficiency of these attacking methods. All of these methods are expected to decrease the classification accuracy, while the PWWS should reduce that to the most extent. We will compare our results with the paper to check the functionality of our replication. Additionally, we will also conduct further studies about transferability [2], [3] and adversarial training [4]. We hope these experiments can help to improve the robustness of the NLP models.

2 Related Work

Among the research on the adversarial attacking methods in NLP, we focus on the related work under a black-box setting, where we do not have access to the input features and the model's architecture. Meanwhile, we also want to ensure that the generated examples do not contain any lexical or grammar errors. Based on the above criteria, we choose the following four attacking methods as the baselines for our project.

- Random [1]: randomly selecting a synonym for each word in the original input text to substitute until the model produces a different result.
- Gradient [2]: using gradient cost function and choosing a synonym that maximizes the change of the prediction output.
- Traversing in word order (TiWO) [1]: traversing the input in the word order and finding a substitute for each word to maximize the drop in the classification probability.
- Word Saliency (WS) [5]: sorting words based on the word saliency in increasing order and finding a substitute for each word according to the maximum likelihood estimation.

Dataset	#Classes	#Train samples	#Test samples	#Avg. words	Task
IMDB Review	2	25,000	25,000	325.6	Sentiment analysis
AG's News	4	120,000	7600	278.6	News categorization
Yahoo! Answers	10	1,400,000	50,000	108.4	Topic classification

Table 1: Statistics of the datasets. #Avg. words is the average number of words in each sentence.

3 Datasets

We will use the same datasets as the paper, including IMDB [6], AG's News [7], and Yahoo! Answers [7]. Table 1 lists the details of each dataset. We will evaluate the attacking methods based on the decreased amount of models' accuracy and the word substitution rate. The most efficient method should reduce the accuracy to the most extent with a relatively low word substitution rate. If time allows, we will try different datasets to see if different tasks and word domains can significantly change the adversary methods' efficiency.

4 Tasks and Resources

We will implement the PWWS and evaluate its performance by attacking several classic and state-of-the-art models for text classification. These models include the word-based CNN [8], the LSTM [9], and the bi-directional LSTM [10], [11]. We will first construct these models and train them on the datasets shown in Table 1. After that, we will reproduce PWWS and other baselines to generate adversarial examples and attack the above models. All of these models and attacking methods will be implemented using PyTorch. We may also use other common libraries, such as NLTK[12], Numpy[13], and pandas[14] for the data preprocess and the result analysis. We estimate that the whole project may require around 30 computational hours for model training and adversarial example generation.

5 Milestones

The timeline of the project is shown in the following. We will draft the project report after each experiment and finalize it in the last step.

- Oct. 20 - 31: Constructing and training models for text classification.
- Nov. 1 - 6: Implementing the PWWS and evaluating its performance.
- Nov. 7 - 13: Implementing baseline methods and and evaluating their performance.
- Nov. 14 - 19: Analyzing the preliminary results and preparing for the presentation.
- Nov. 20 - 31: Trying to improve the PWWS and conducting further analysis.
- Dec. 1 - 16: Finalizing the whole project and the report.

6 Team Breakdown

We will construct and train NLP models jointly. For the attacking methods, Jinze will focus on the baselines while Fengyi will implement the PWWS. We will analyze the experiments and prepare for the presentation and the final report together.

References

[1] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 1085–1097.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[3] C. Szegedy, W. Zaremba, I. Sutskever, *et al.*, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[4] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2107–2116.

[5] S. Samanta and S. Mehta, "Towards crafting text adversarial samples," *arXiv preprint arXiv:1707.02812*, 2017.

[6] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150. [Online]. Available: `http://www.aclweb.org/anthology/P11-1015`.

[7] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *NIPS*, 2015.

[8] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: `10.3115/v1/D14-1181`. [Online]. Available: `https://aclanthology.org/D14-1181`.

[9] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015.

[10] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," *arXiv preprint arXiv:1611.06639*, 2016.

[11] G. Liu and J. Guo, "Bidirectional lstm with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, pp. 325–338, 2019.

[12] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[13] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: `10.1038/s41586-020-2649-2`. [Online]. Available: `https://doi.org/10.1038/s41586-020-2649-2`.

[14] T. pandas development team, *Pandas-dev/pandas: Pandas*, version latest, Feb. 2020. DOI: `10.5281/zenodo.3509134`. [Online]. Available: `https://doi.org/10.5281/zenodo.3509134`.